New-York City Real-Estate Price Predictor

Nirmal Sai Swaroop Janapaneedi

5/20/2020

Introduction

In this project I will create a price prediction algorithm for New-York City real-estate prices for the final project of the course HarvardX: PH125.9x - Data Science: Capstone. The algorithm is trained based on New-York city property sales, imported from kaggle website. The method chosen for training the algorithm is random forest.

The kaggle website describes the data as a year's worth of properties sold on the NYC real estate market. The dataset is a record of every building or building unit (apartment, etc.) sold in the New York City property market over a 12-month period. The data contains transactions made starting on September 1st, 2016 and finishing on August 31, 2017. The first section will explore and describe the data; The Methods&Analysis section will study the algorithm used, training it on the train set; the Results section will test the algorithm on the test set and the conclusions part will discuss the results and suggest further analysis.

Exploring the data

The data contains the information about 84,548 real-estate transactions made in New-York City starting on September 1st, 2016 and finishing on August 31, 2017.

```
## # A tibble: 6 x 22
##      X1 BOROUGH NEIGHBORHOOD `BUILDING CLASS~ `TAX CLASS AT P~ BLOCK   LOT
##   <dbl>   <dbl> <chr>        <chr>            <chr>            <dbl> <dbl>
## 1     4       1 ALPHABET CI~ 07 RENTALS - WA~ 2A                 392     6
## 2     5       1 ALPHABET CI~ 07 RENTALS - WA~ 2                  399    26
## 3     6       1 ALPHABET CI~ 07 RENTALS - WA~ 2                  399    39
## 4     7       1 ALPHABET CI~ 07 RENTALS - WA~ 2B                 402    21
## 5     8       1 ALPHABET CI~ 07 RENTALS - WA~ 2A                 404    55
## 6     9       1 ALPHABET CI~ 07 RENTALS - WA~ 2                  405    16
## # ... with 15 more variables: `EASE-MENT` <lgl>, `BUILDING CLASS AT
## #   PRESENT` <chr>, ADDRESS <chr>, `APARTMENT NUMBER` <chr>, `ZIP CODE`
<dbl>,
## #   `RESIDENTIAL UNITS` <dbl>, `COMMERCIAL UNITS` <dbl>, `TOTAL UNITS`
<dbl>,
## #   `LAND SQUARE FEET` <chr>, `GROSS SQUARE FEET` <chr>, `YEAR BUILT`
<dbl>,
## #   `TAX CLASS AT TIME OF SALE` <dbl>, `BUILDING CLASS AT TIME OF SALE`
```

```
<chr>,
## #   `SALE PRICE` <chr>,  `SALE DATE` <dttm>
```

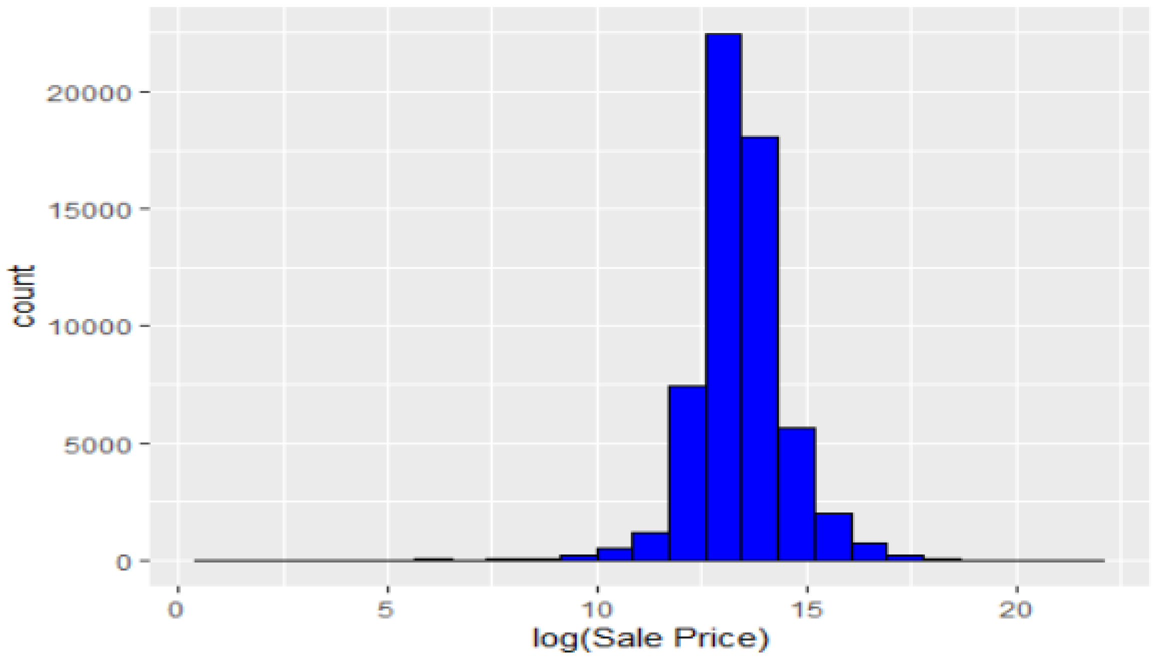
The different variables are:

```
##   [1]  "X1"                              "BOROUGH"
##   [3]  "NEIGHBORHOOD"                    "BUILDING CLASS CATEGORY"
##   [5]  "TAX CLASS AT PRESENT"            "BLOCK"
##   [7]  "LOT"                             "EASE-MENT"
##   [9]  "BUILDING CLASS AT PRESENT"       "ADDRESS"
##  [11]  "APARTMENT NUMBER"                "ZIP CODE"
##  [13]  "RESIDENTIAL UNITS"               "COMMERCIAL UNITS"
##  [15]  "TOTAL UNITS"                     "LAND SQUARE FEET"
##  [17]  "GROSS SQUARE FEET"               "YEAR BUILT"
##  [19]  "TAX CLASS AT TIME OF SALE"       "BUILDING CLASS AT TIME OF SALE"
##  [21]  "SALE PRICE"                      "SALE DATE"
```

The glossary of terms describes the meaning of each one of the variables.
Some basic exploration of the data shows that the sale price of some of the transactions is lower than $1,000. This might be due inheritence, non-monetary exchange, etc. I have filtered the transactions with a sale price lower than $1,000 for a more presice analysis.

```
## # A tibble:   2 x 2
##    `prices$\`SALE   PRICE\` < 1000`       n
##    <lgl>                              <int>
## 1 FALSE                               58769
## 2 TRUE                                25779
```

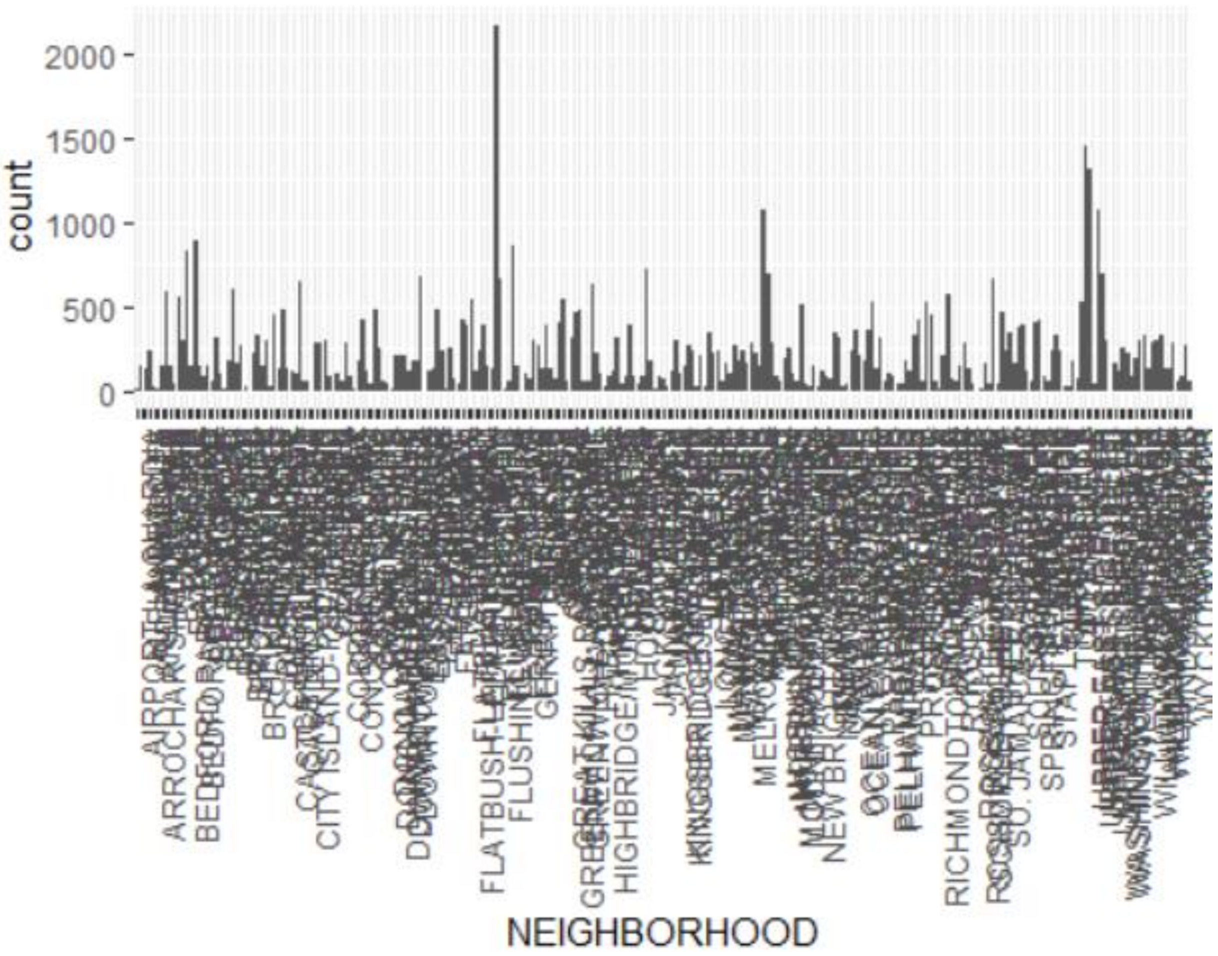
A log transformation of the sale price shows a somewhat normal distribution of the sale prices.



I will return to the sale price later for further analysis, but first, let's examine the other variables. The data includes the Building Class Category. The building class category, explained by the glossary of terms as a field used to describe the property's broad usage.

```
## # A tibble: 46 x 2
##    `BUILDING CLASS CATEGORY`          n
##         <chr>                       <int>
##   1 01 ONE FAMILY DWELLINGS         12720
##   2 02 TWO FAMILY DWELLINGS          9886
##   3 03 THREE FAMILY DWELLINGS        2326
##   4 04 TAX CLASS 1 CONDOS            1247
##   5 05 TAX CLASS 1 VACANT LAND        497
##   6 06 TAX CLASS 1 - OTHER             48
##   7 07 RENTALS - WALKUP APARTMENTS   1750
##   8 08 RENTALS - ELEVATOR APARTMENTS  199
##   9 09 COOPS - WALKUP APARTMENTS     2505
## 10 10 COOPS - ELEVATOR APARTMENTS   11522
## # ... with 36 more rows
```

The data contains information about the neighberhood of the property. Some neighborhoods have more transactions than others.

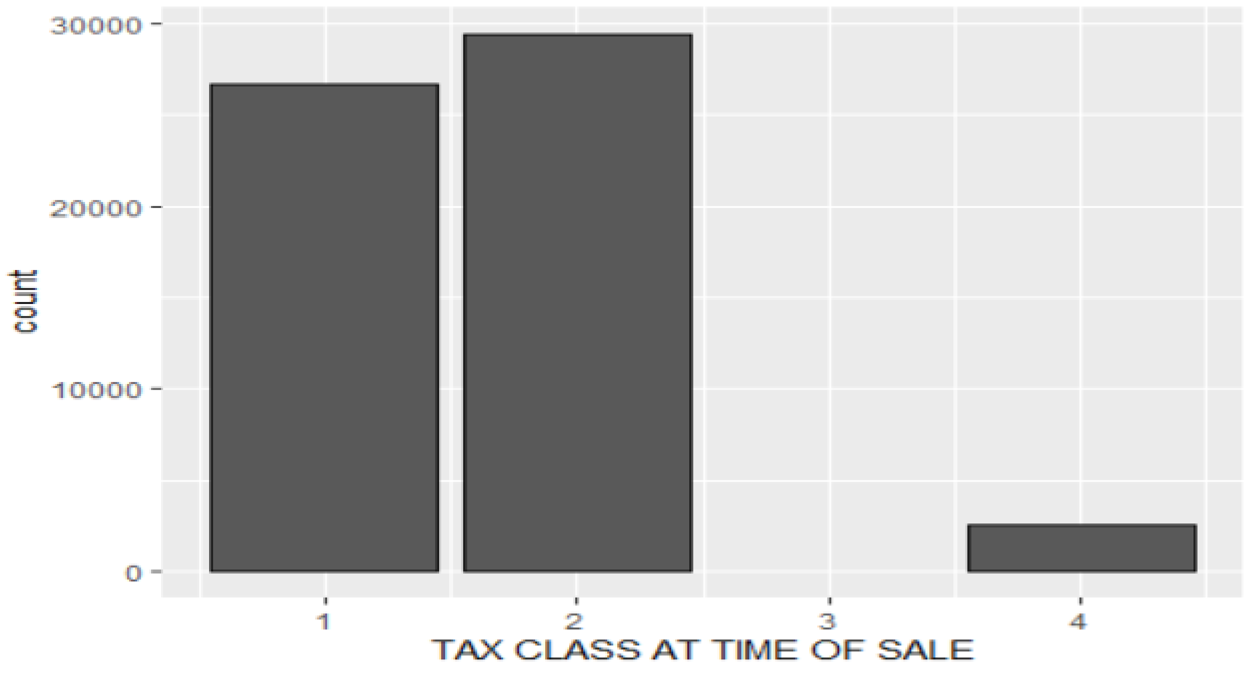


```
## # A tibble: 10 x 3
##    NEIGHBORHOOD                  n mean_sale_price
##    <chr>                     <int>           <dbl>
##  1 FLUSHING-NORTH             2165            989.
##  2 UPPER EAST SIDE (59-79)    1459           2445.
##  3 UPPER EAST SIDE (79-96)    1312           2359.
##  4 UPPER WEST SIDE (59-79)    1078           2494.
##  5 MIDTOWN EAST               1066           1868.
##  6 BEDFORD STUYVESANT          887           1331.
##  7 FOREST HILLS                864            550.
##  8 BAYSIDE                     836            693.
##  9 JACKSON HEIGHTS             723            718.
## 10 UPPER WEST SIDE (79-96)     701           2321.
```

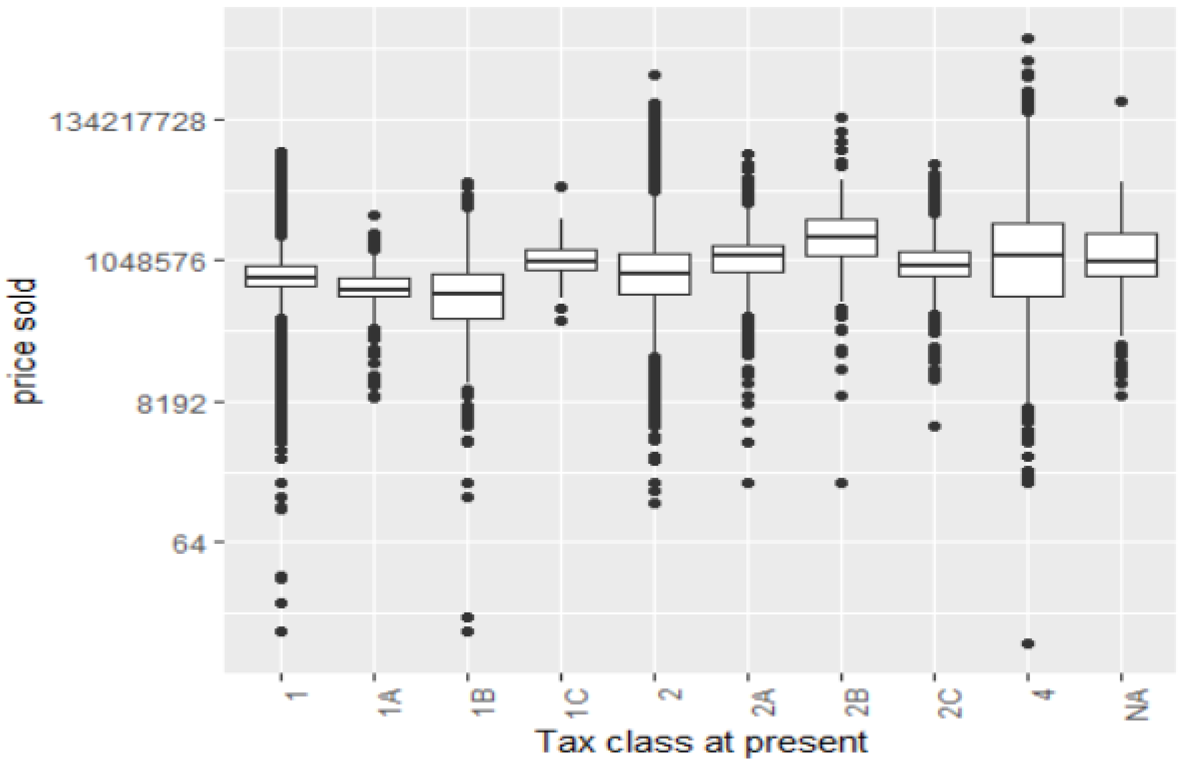
Every property in the city is assigned to one of four tax classes, with tax class 1 and 2 including residential properties, 3 property with equipment owned by gas, telephone or electric companies and 4 all other properties (offices, factories, warehouses, garage buildings, etc.).

```
## # A tibble: 3 x 3
##   `prices$\`TAX CLASS AT TIME OF SALE\`` mean_sale_price     n
##                                    <dbl>           <dbl> <int>
## 1                                      1            743. 26724
## 2                                      2           1611. 29438
## 3                                      4           8716.  2530
```

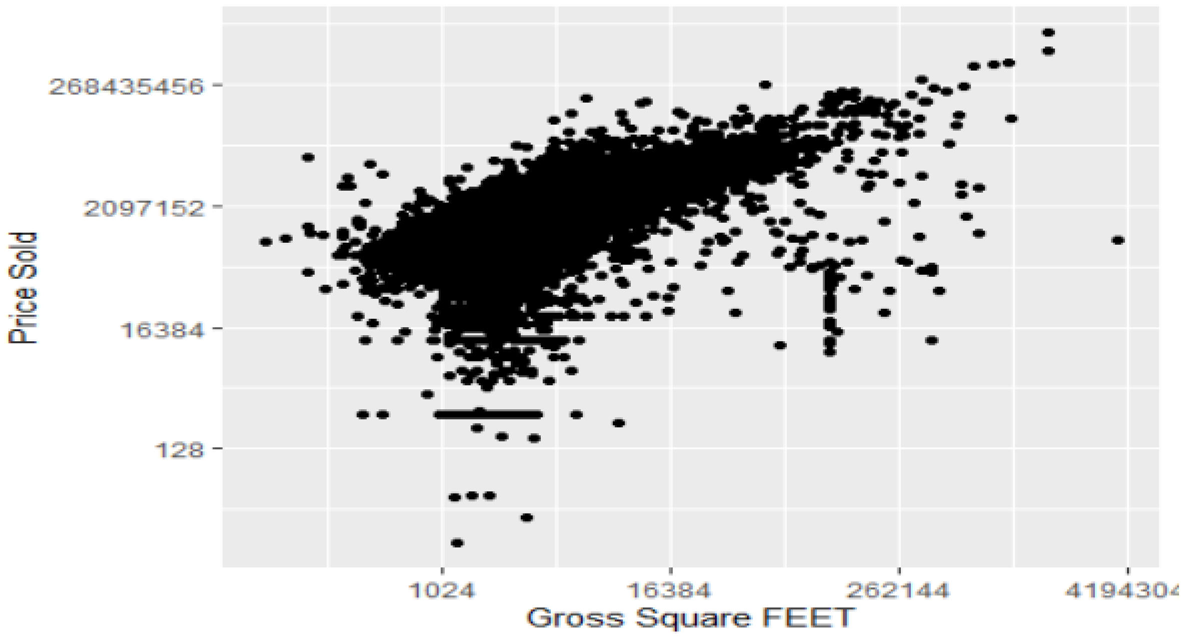
We can see that the majority of properties sold are in class 1 and 2. The mean sale price is much higher for tax class 4 properties.



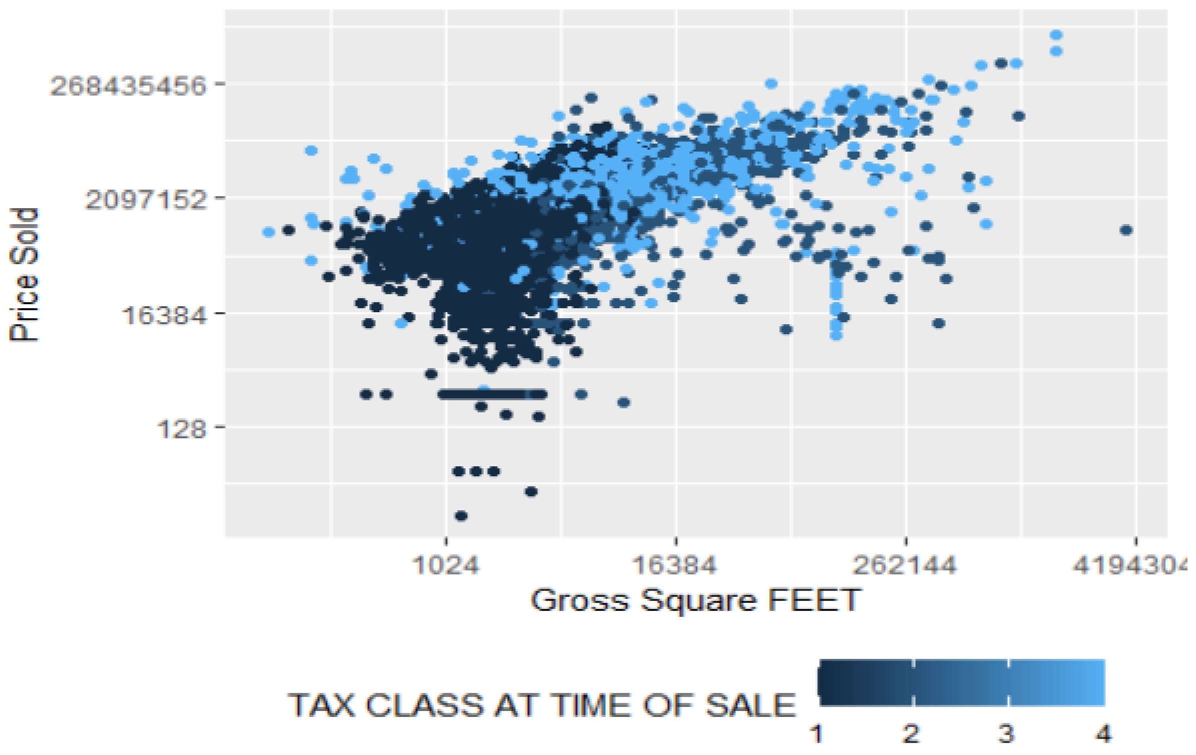
We can see that also among the tax classes at present, there is some variance and some extra information on the different tax classes.



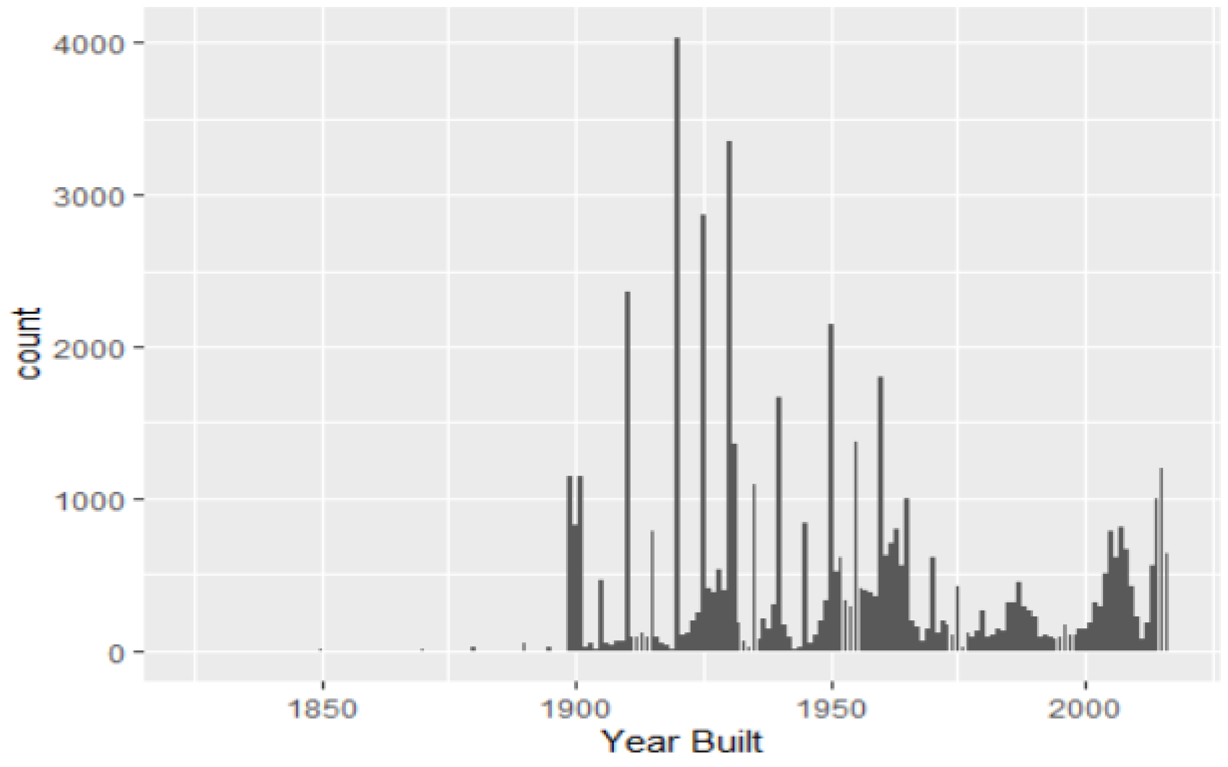
The data includes information about the gross square feet of the property. The gross square feet is not included for all the properties. Filtering the NA’s, shows some connection between sale price and the size of the property.



Adding the tax class to the graph shows that tax class adds some more information:

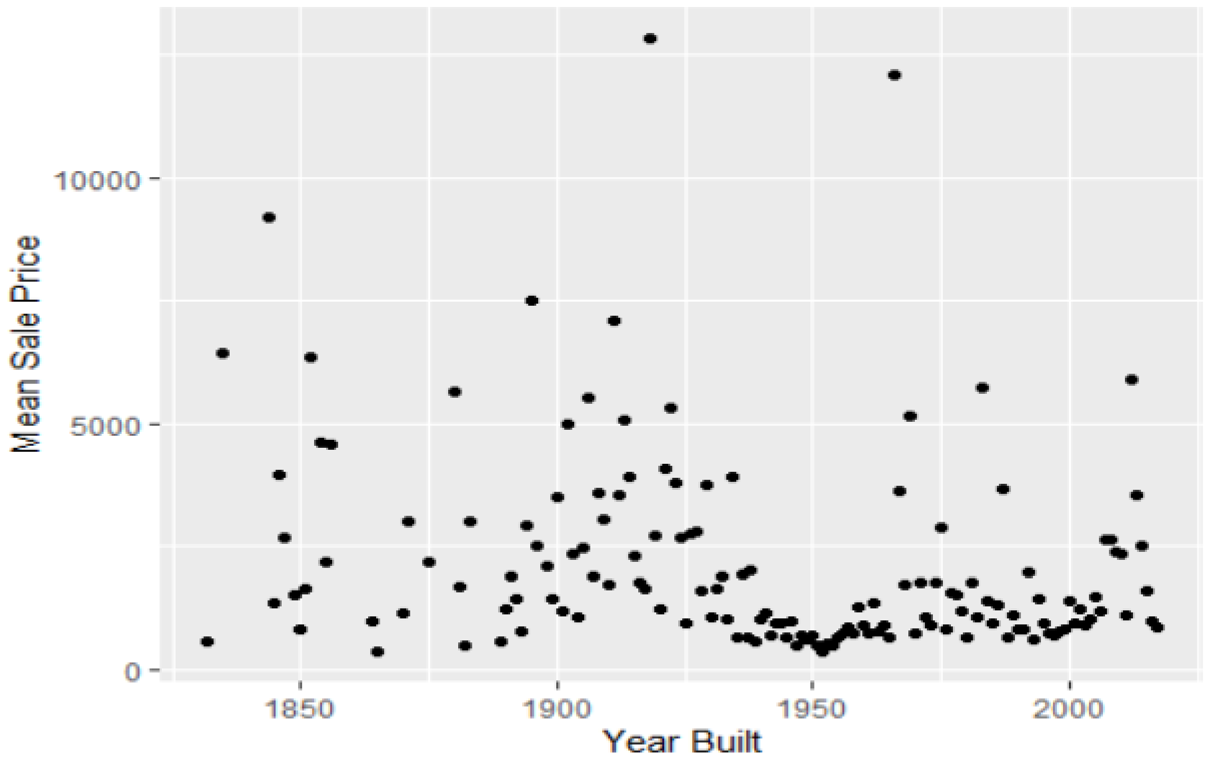


The data also includes information about the construction year of the property. We can see that for some of the years constructed there were more transcations in the data.



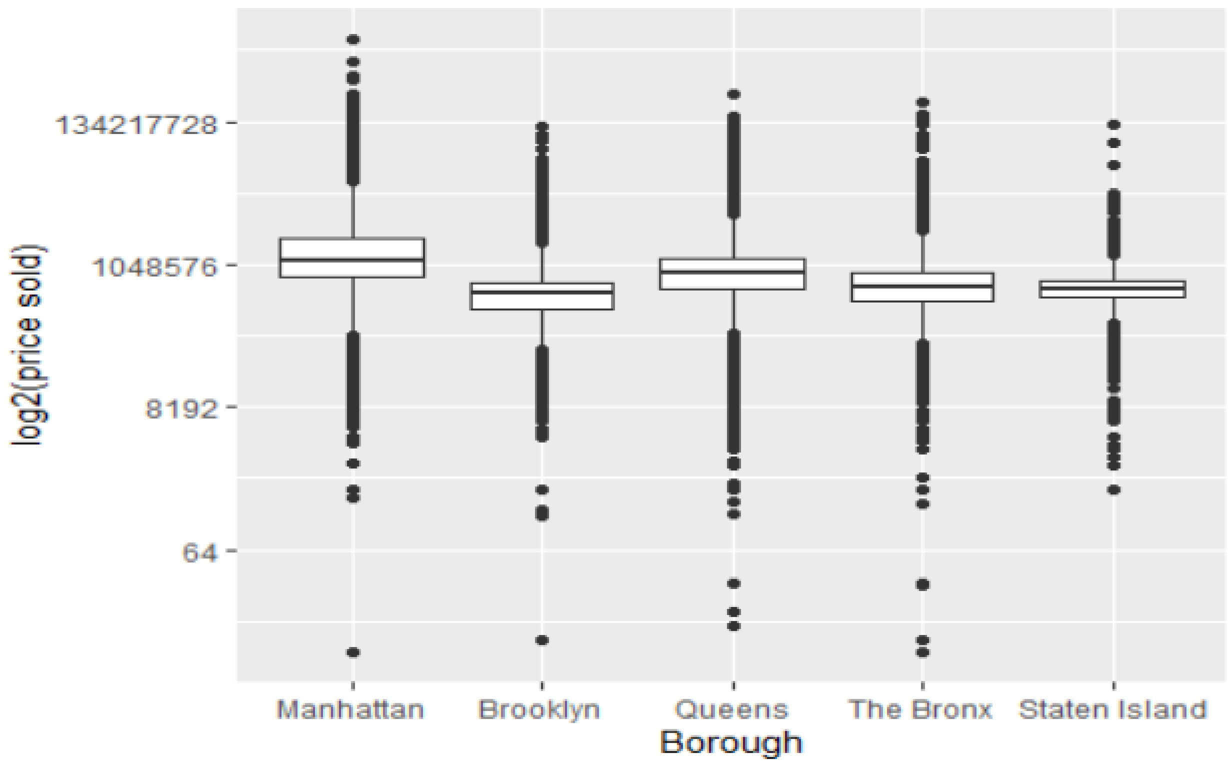
We can also see some possible connection between the construction year and the mean sale price.

```
## # A tibble: 10 x 3
##   `YEAR BUILT` mean_sale_price     n
##          <dbl>           <dbl> <int>
##  1        1832             585     1
##  2        1835            6442.    2
##  3        1844            9200     2
##  4        1845            1337.    4
##  5        1846            3950     2
##  6        1847            2675     1
##  7        1849            1520     1
##  8        1850             825.    5
##  9        1851            1638.    2
## 10        1852            6375     2
```

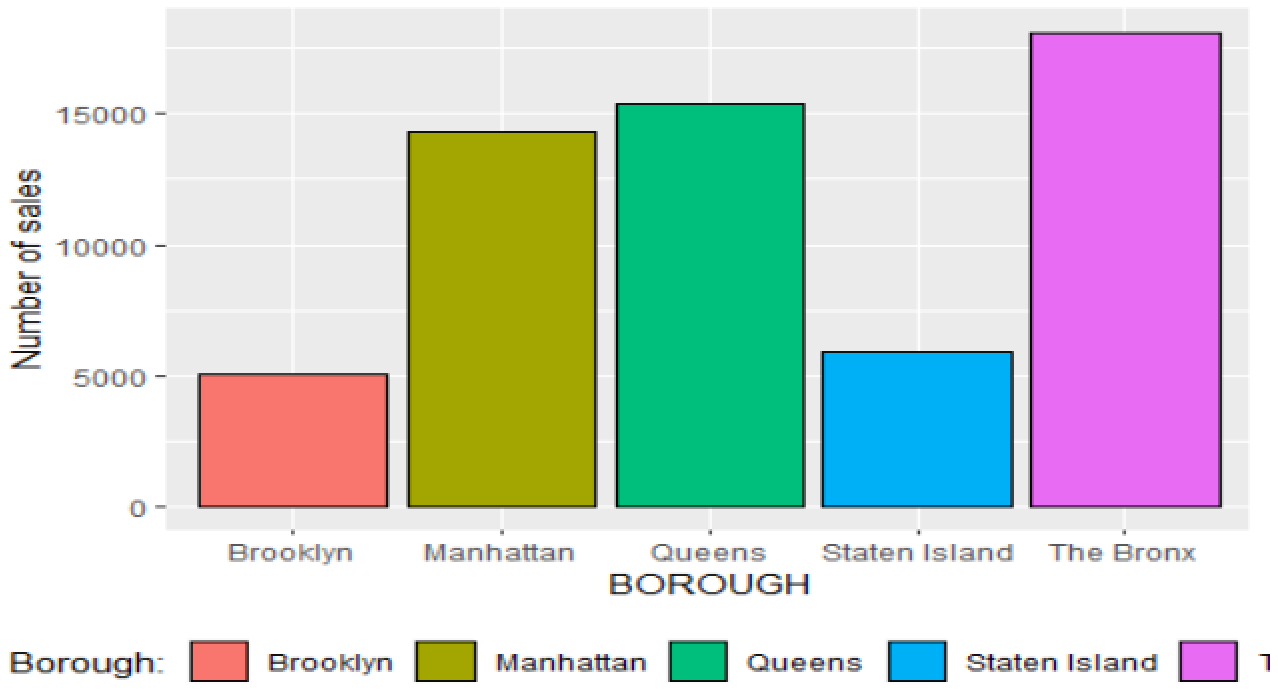


The city of New-York is divided into five boroughs: Manhattan, Brooklyn, Queens, The Bronx and Staten Island. The data includes the borough information.

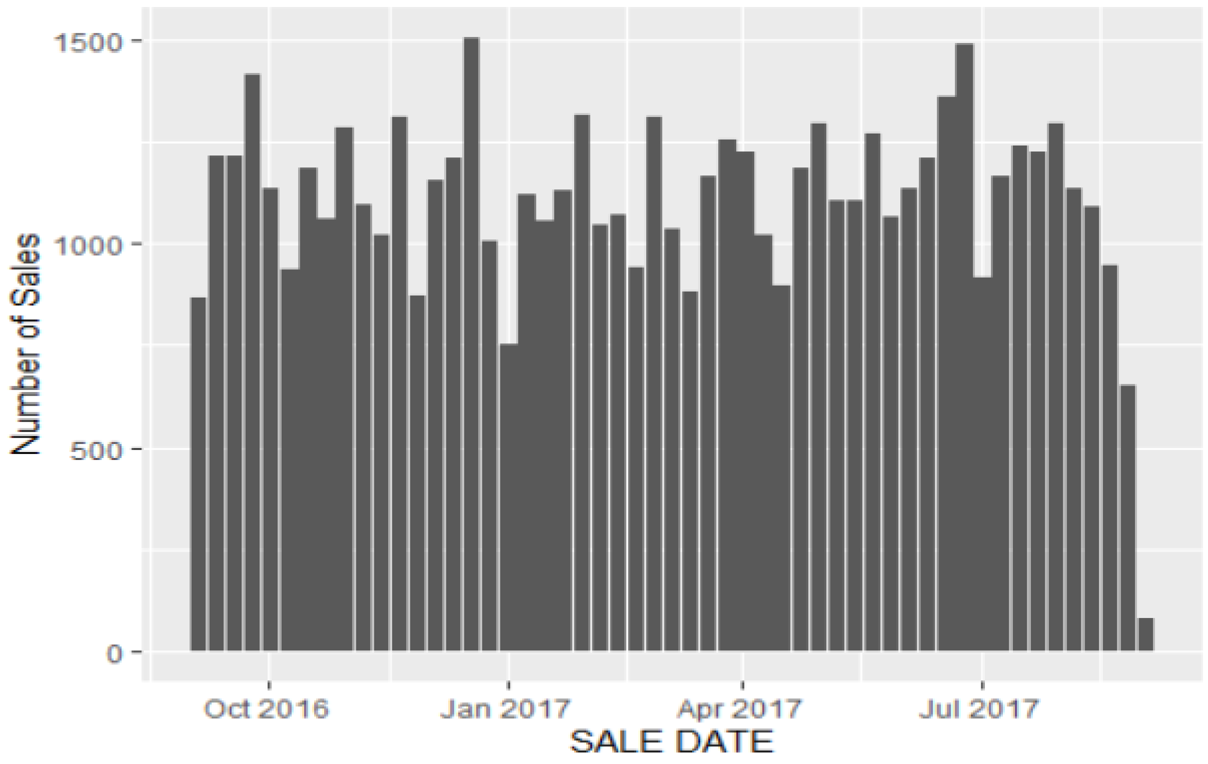
```
## # A tibble: 5 x 3
##   BOROUGH      mean_sale_price     n
##     <chr>                <dbl> <int>
## 1 Brooklyn                827.  5030
## 2 Manhattan              3369. 14305
## 3 Queens                 1307. 15353
## 4 Staten   Island                    555.    5883
## 5 The  Bronx                         753.   18121
```



We can see that The Bronx had the most transactions, yet Manhattan had the highest mean sale price.



The date of the sale is also included in the data and might be correlated with the price. Grouping the sales by weeks, we can see that some weeks had more transactions than others.



I have tidied the data, changing the the variable names and coercing some of them to numerical and factor variables for further analysis.

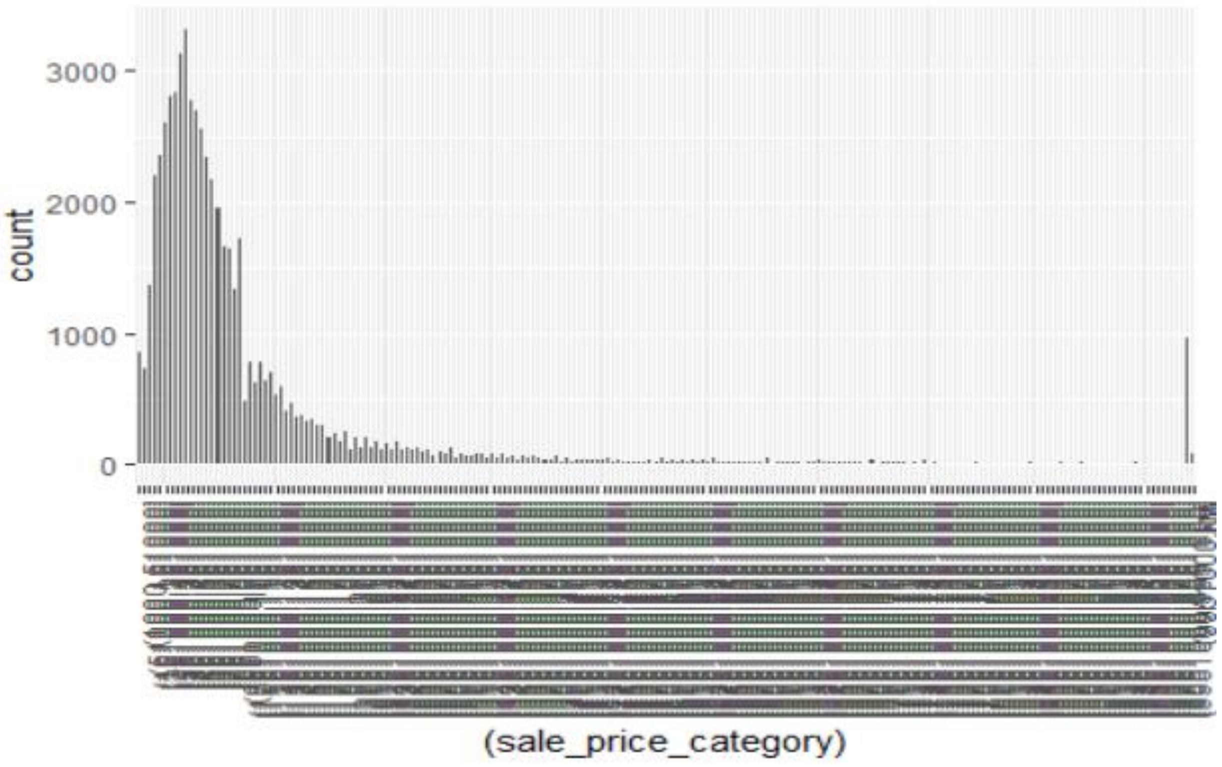
Methods and Analysis

Data Preparation

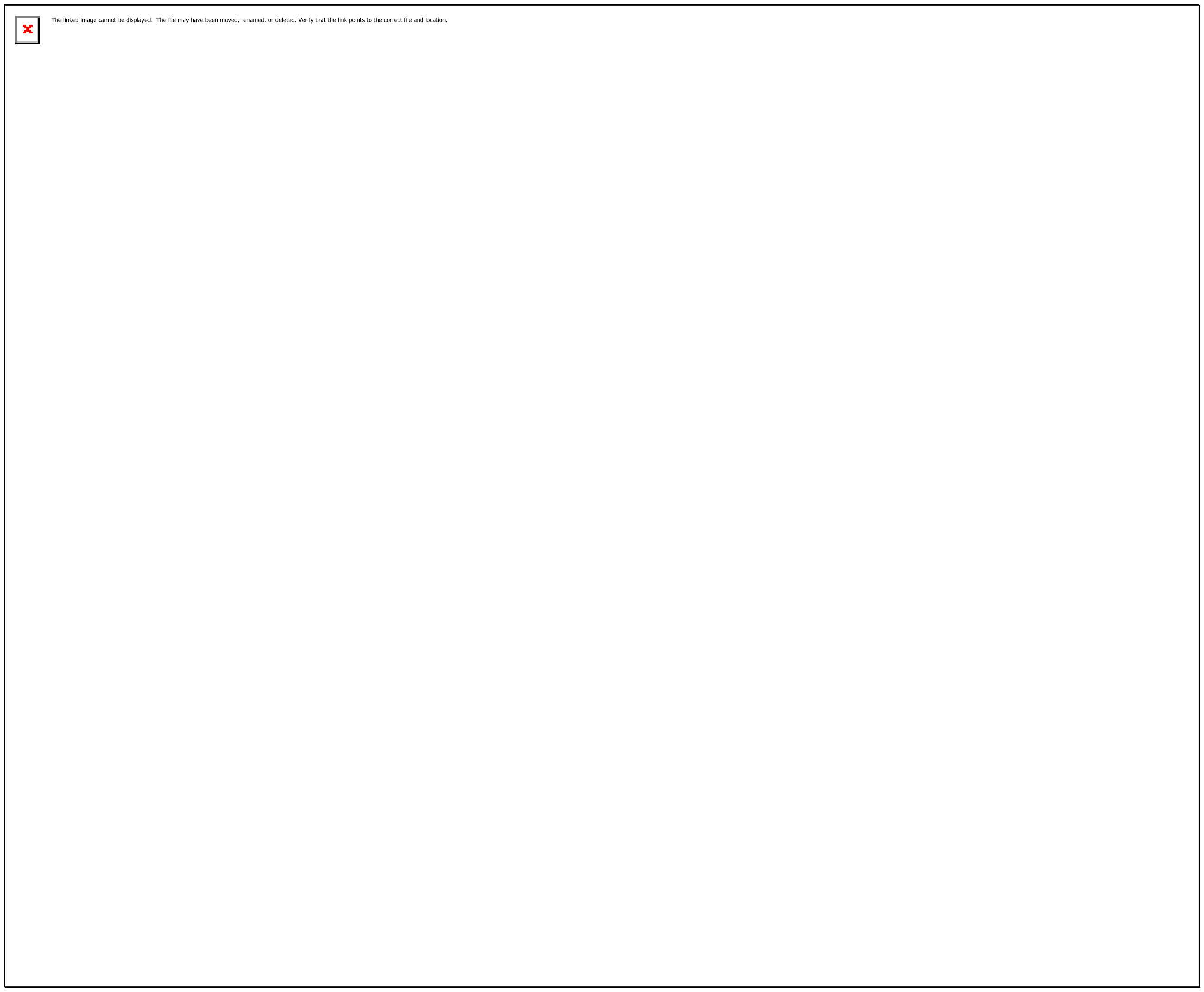
The first method which I have attempted was a general linear model, using the caret package. The computing time for that model and other models on the caret package was often longer than 12h and did not provide a satisfying prediction. I have therefore, divided the data into categories, with the intention of predicting the price category of the property. I have attempted several different category borders, eventually using $50,000 as the group size. The observations have been divided into equal sized borders with $1,000 being the lowest limit, and the last group includes all values above $10,000,000.

```
## # A tibble:     10 x 2
##      sale_price_category           n
##      <fct>                     <int>
##   1  (1000,51000]                853
##   2  (51000,101000]              731
##   3  (101000,151000]            1366
##   4  (151000,201000]            2204
##   5  (201000,251000]            2356
##   6  (251000,301000]            2603
##   7  (301000,351000]            2801
##   8  (351000,401000]            2826
##   9  (401000,451000]            3124
##  10  (451000,501000]            3319
```

We see that the sale price categories distribution follows a somewhat right tail x distribution, with an extremely large group for value prices of $9,950,000 to $10,000,000.



Before going into training the algorithm, I examined the correlataions between the different varaibles and have removed variables with very high correlation (close to 1). A heatmap of the correlations between the different variables shows that some variables are more correlated to the sale price category than others. I have omitted the na’s from the correlation calculation.


The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

We can see that some of the variables have NA’s.

```
##                neighborhood   building_class_at_time_of_sale
##                           0                                0
##      tax_class_at_time_of_sale                     year_built
##                           0                                0
##                    gross_sf                          land_sf
##                       21569                            21033
##                 total_units                 commercial_units
##                           0                                0
##           residential_units                         zip_code
##                           0                                0
##            apartment_number                              lot
##                       45350                                0
##                     borough          building_class_category
##                           0                                0
##         tax_class_at_present                            block
##                         593                                0
```

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

```
##        building_class_at_present                              address
##                               593                                    0
##                         sale_date              sale_price_in_thousands
##                                 0                                    0
##                        sale_price                  sale_price_category
##                                 0                                   88
```

Random Forest Algorithm

As I mentioned, my first attempts of creating an algorithm included using the train function in the caret package. Using my dual-core 16GB laptop took well over 15 hours and therefore I have looked for different options, eventually using the ranger for implementing a random forest algorithm. The ranger package manual describes the package as “A fast implementation of Random Forests”, and indeed, computing time has improved drastically. I divided the data into train and test sets, trained the algorithm on the train set and tested it on the test, reporting the evaluation metrics described in the following section. I have also filtered some na colomns (correlated variables).

Evaluation Metrics

In order to evaluate the different algorithms, originally, three evaluation metrics were used. The different evaluation metrics were calculated on the test set.

1. Root Mean Square Error (RMSE):

$$RMSE=\sqrt{\frac{1}{N}\sum(r_{i,j}-\hat{r}_{i,j})^2}$$

2. Mean Average Error:

$$MAE=\sum(|r_{i,j}-\hat{r}_{i,j}|)$$

3. Mean Square Error:

$$MSE=\frac{1}{N}\sum(r_{i,j}-\hat{r}_{i,j})^2$$

Since my original implementation included long computing time and resulted in unsatisfying results, I have reported the metrics just for the final algorithm. Since those metrics are used for comparing different algorithms, some sort of an abosolute metrics had to be thought of. I have decided to use the percentage of correct category prediction as a final metric and reporting it in addition to the other mertrics described.

Results

The random forest algorithm predicts quite accurately the sale price category. Applying it to the test set results in the following metrics:

```
## # A tibble:   1 x 4
##    method                                    RMSE     MSE     MAE
```

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

```
##   <chr>                                          <dbl>  <dbl>  <dbl>
## 1 Random  Forest  using  ranger  libary          5.94   35.3  0.908
```

The percentage metrics calculated is

```
## [1]  0.9119
```

Thus, satisfying the scopes of this project and creating a comfortable tool for prediction.

Conclusion

In this project I created a price prediction algorithm for New-York City real-estate, using the kaggle database. The main disadvantage of my project is my weak laptop, not being able to process some possible algorithms, and therefore I did not compare my algorithm to any other algorithms but only reached what is, in my eyes, a satisfying result. In the future, and with a faster laptop I would try implemenging some other algorithms and also add some external variables, such as GDP, stock exchange prices, construction prices etc.